IBM

Rational

Functional

Tester

功功能测试基本使用手册

版本1.0

目录：

第一节：Rational Functional Tester产品介绍 5

* 1. Rational Functional Tester概述 5
  2. Rational Functional Tester基本特性 5

1.3RationalFunctionalTester 基本特性 5

1.3.1回放更新的应用程序脚本 5

1.3.2更新对象的识别属性 6

1.3.3合并多个测试对象地图 6

1.3.4现实相关脚本 6

1.3.5使用基于模式的对象识别 6

1.3.6集成UCM 7

第二节：Rational Functional Tester产品展示 8

2.1用户界面 8

2.1.1主菜单 8

2.1.2工具栏 8

2.1.3Functional Test Projects试图 10

2.1.4Java编辑器 11

2.1.5Script Explorer 12

2.1.6Console试图 12

2.1.7Tasks试图 12

2.1.8状态栏 13

第三节Rational Functional Tester产品使用 14

3.1入门 14

3.1.1启动RFT 14

3.2准备测试环境 15

3.2.1新建功能测试项目 14

3.2.2启用应用程序环境 15

3.2.3配置被测试应用程序 16

3.3录制测试 16

3.3.1启动录制器 17

3.3.2启动被测试的应用程序 18

3.3.3创建数据驱动的订单 18

3.3.4验证动态数据 20

3.3.5验证静态数据 24

3.3.6停住应用程序并停止录制 25

3.4检验测试 25

3.4.1检验测试对象图(Test Object Map) 25

3.4.2向数据池中添加数据变更 26

3.4.3准备ClassicsJava Build B 29

3.5运行自动化回归测试 29

3.5.1执行测试 29

3.6分析测试日志中的结果 30

3.6.1ScriptAssure 30

3.6.2定制日志信息 31

3.6.3研究验证点状态 31

3.6.4验证数据池操作 32

3.7总结 32

第四节Rational Functional Tester产品使用 33

4.1关于开发者测试的公共问题 33

4.2 Rational Functional Tester 34

4.3超过简单的断言 35

第五节Rational Functional Tester产品使用 36

4.3学习 36

4.3获得产品和技术 37

4.3其他可参考学习地址 37

# 第一节：Rational Functional Tester产品介绍

### 1.1 Rational Functional Tester概述

Rational Functional Tester（以下简称RFT）是一个面向对象的、自动测试工具，它使您能够测试各种应用程序。通过录制一个应用程序的测试您可以很快产生测试脚本，您还可以测试这个应用程序之中的任何对象，包括这个对象的属性和数据。RFT可以给您提供一个编写脚本语言的机会和两种开发环境：Eclipse 框架中的 Java™或者 Microsoft™ Visual Studio™开发系统中的 Microsoft™ Visual Basic™ .NET。这意味着，无论您开发小组的成员使用什么样的语言或者平台，您都应该能够将它们与 Rational Functional Tester 集成起来，并且您在开发自动化测试的时候能够利用它们的一些功能。

可查看：<http://tech.ddvip.com/2008-12/122857114098145.html>

### 1.2 Rational Functional Tester基本特性

RFT的基础是针对于Java、.NET的对象技术和基于 Web 应用程序的录制、回放功能。工具为测试者的活动提供的自动化的帮助，如数据驱动测试。

当您记录脚本时，RFT会为被测的应用程序自动创建测试对象地图。对象地图中包含了对每个对象的识别属性。当您在对象地图中更新记录信息时，任何使用了该对象地图的脚本会共享更新的信息，减少了维护的成本及整个脚本开发的复杂度。对象地图还为您提供快速的方法向脚本中添加对象。它列出应用程序中涉及到的测试对象，不论它们当前是否可视。您可以通过依据现有地图或按需添加对象来创建新的测试对象地图。

在记录过程中您可以将验证点插入到脚本中以确定在被测应用程序建立过程中对象的状态。验证点获取对象信息（根据验证点的类型，可以是对象属性验证点或五种数据验证点之一 -- 菜单层次、表格、文本、树形层次，或列表）并在基本数据文件中存储。文件中的信息成为随后的建立过程中对象的期望状态。在执行完测试之后，您可以使用验证点比较器（Verification Point Comparator） 进行分析，并且如果对象的行为变化了就更新基线（期望的对象状态）。

可参考：<http://www.ibm.com/developerworks/cn/rational/r-rft/index1.html>

## 1.3 Rational Functional Tester功能

##### 1.3.1 回放更新的应用程序脚本

ScriptAssure 特性是 Rational Functional Tester 的对象识别技术，可以使您成功地回放脚本，甚至在您的被测应用程序已经更新的时候。您可以为测试对象必须通过的、用来作为识别候选的识别记分设置门槛，并且如果 Rational Functional Tester 接受了一个分值高于指定门槛的候选，您还可以向日志文件中写入警告。

##### 1.3.2 更新对象的识别属性

在测试对象地图中，您可以对所选的测试对象更新识别到的属性。Rational Functional Tester 显示了 Update Recognition Properties 页，其显示出更新的测试对象属性、原始的识别属性和对象所有的识别属性。如果必要，您可以修改更新的识别属性。

##### 1.3.3 合并多个测试对象地图

对象地图要么是共享的要么是专用的。专用地图附属于一个脚本并只由具体的脚本进行访问，反之，共享的地图由多个脚本共享。共享地图的优势是，当需要更新对象时，只有对应一个地图的一个更新会确定多个脚本。您可以在RFT的项目视图中并且在创建新测试对象地图时，将多个私有的或共享的测试对象地图合并成一个单个的共享测试对象地图。RFT可以随意地更新您所选择的指向新合并的测试对象地图的脚本。

##### 1.3.4 显示相关的脚本

在测试对象地图中，您可以观察到与列表与地图相关的脚本，且可以使用该列表来选择要添加测试对象的多个脚本。

##### 1.3.5 使用基于模式的对象识别

您可以用正则表达式或一个数值范围来代替允许基于模式的识别。允许对象识别具有更好的灵活性。您可以将属性转变成验证点编辑器（Verification Point Editor） 或测试对象地图中的正则表达式和数值范围。正则表达式计算器（Regular Expression Evaluator）允许您在编辑表达式时进行测试，这节省下了不得不运行脚本观察模式是否工作的时间。

##### 1.3.6 集成UCM

RFT在ClearCase统一变更管理（Unified Change Management，UCM）的视图中。RFT中创建的工件是可以进行版本控制的。

可参考<http://www.ibm.com/developerworks/cn/rational/r-rft/index1.html>

# 第二节：Rational Functional Tester产品展示

在本课程中我们将集中使用Eclipse框架中的Java的Rational Functional Tester。本章节将为您展示RFT的用户界面。

### 2.1用户界面

当您启动 Rational Functional Tester For Java 时，您会看到带有八个主要组件的 Test Perspective 窗口：主菜单、工具栏、Project 视图、Java 编辑器、Script Explorer、Console 视图、Tasks 视图和状态栏。下面是对每个组件的简要描述。

##### 2.1.1 主菜单

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image001.jpg

您可以在 Rational Functional Tester 的在线帮助中读到关于主菜单中每个选项的内容。

##### 2.1.2 工具栏

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image002.jpg

工具栏中包含这些图标：

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image003.jpgOpen the New Wizard -- 显示适当的对话框来创建许多项中的一个或录制 Functional Test 脚本。单击 http://www.ibm.com/developerworks/cn/rational/r-rft/images/image006.jpg以显示要创建的可能项列表。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image007.jpgCreate New Functional Test Project -- 显示出一个对话框，让您在 Functional Test 中生成新工程。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image008.jpgConnect to an Existing Functional Test Project -- 显示出一个让您连接到现有工程的对话框。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image009.jpgCreate an Empty Functional Test Script -- 显示出一个让您创建您可以用来手动地添加 Java 代码的脚本的对话框。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image011.jpgCreate New Test Object Map -- 显示出一个让您向工程添加一个新的测试对象地图的对话框。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image012.jpgCreate New Test Datapool -- 显示出一个让您创建一个新的测试数据库的对话框。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image014.jpgCreate a New Test Folder -- 显示出一个让您为工程或现有文件夹创建一个新文件夹的对话框。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image015.jpgRecord a Functional Test Script -- 显示出一个让您输入关于新脚本的信息并开始记录的对话框。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image016.jpgInsert Recording into Active Functional Test Script -- 在当前脚本的光标位置开始记录，这使您启动应用程序、插入验证点，并添加脚本支持功能。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image017.jpgConfigure Applications for Testing -- 显示 Application Configuration 工具，其使您添加并编辑配置信息 -- 例如名称、路径和其他用于开始并执行应用程序的信息 -- 关于您要测试的 Java 和 HTML 应用程序。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image018.jpgEnable Environments for Testing -- 显示出一个您用来启动 Java 环境和浏览器及配置 JRE 和浏览器的对话框。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image019.jpgDisplay the TestObject Inspector Tool -- 显示出 TestObject Inspector 工具，使您显示测试对象信息，如父层次、继承层次、测试对象属性、无值属性和方法信息。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image021.jpgInsert Verification Point -- 显示 Verification Point and Action Wizard 的 Select an Object 页，其使您在您要测试的应用程序中选择对象。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image023.jpgInsert Test Object into Active Functional Test Script -- 显示出一个让您选择测试对象来向测试对象地图和脚本中添加的对话框。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image024.jpgInsert Data Driven Commands into Active Functional Test Script -- 显示出 Datapool Population Wizard 的 Data Drive Actions 页，其使您选择被测应用程序中的对象来数据驱动应用程序。

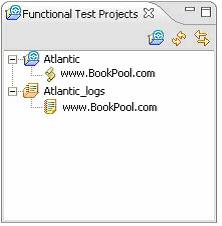
http://www.ibm.com/developerworks/cn/rational/r-rft/images/image025.jpgReplace Literals with Datapool Reference -- 用测试脚本中的数据库参考代替文字值，其使您向现有的测试脚本中添加现实数据。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image026.jpgRun Functional Test Script -- 运行 Functional Test 脚本。单击 http://www.ibm.com/developerworks/cn/rational/r-rft/images/image005.jpg来显示运行命令列表。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image027.jpgDebug Functional Test Script -- 启动当前脚本并显示 Debug Perspective（在脚本调试时提供信息）。单击 http://www.ibm.com/developerworks/cn/rational/r-rft/images/image005.jpg以在当前脚本的方法 Main 中开始调试。单击以显示调试命令列表。

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image028.jpgRun or Configure External Tools -- 使您可以配置非工作台一部分的外部工具。单击 http://www.ibm.com/developerworks/cn/rational/r-rft/images/image005.jpg来显示选项列表。

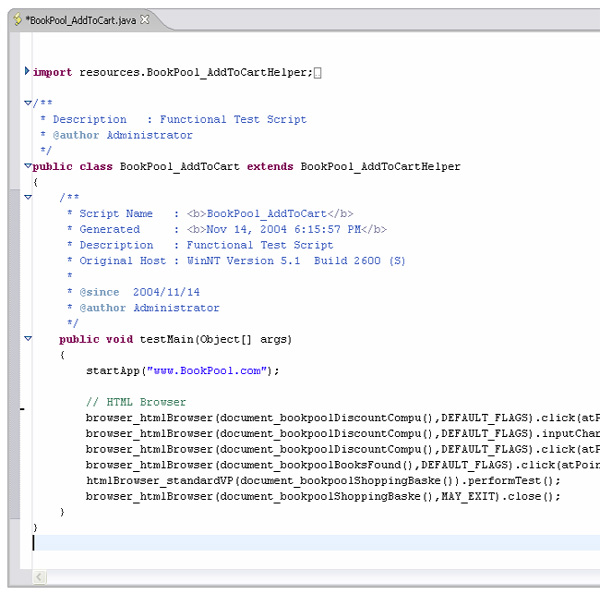
##### 2.1.3 Functional Test Projects视图



Functional Test Projects视图在Test Perspective窗口的左窗格中，为每个工程列出测试资产，包括这些：

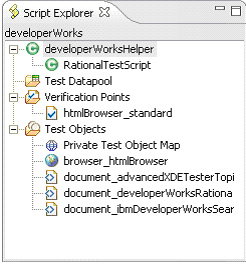
* http://www.ibm.com/developerworks/cn/rational/r-rft/images/image032.jpg文件夹
* http://www.ibm.com/developerworks/cn/rational/r-rft/images/image034.jpg脚本
* http://www.ibm.com/developerworks/cn/rational/r-rft/images/image035.jpg共享的测试对象地图
* http://www.ibm.com/developerworks/cn/rational/r-rft/images/image036.jpg日志文件夹
* http://www.ibm.com/developerworks/cn/rational/r-rft/images/image037.jpg日志
* http://www.ibm.com/developerworks/cn/rational/r-rft/images/image038.jpgJava 文件

##### 2.1.4 Java编辑器



您使用Java编辑器（脚本窗口）编辑Java代码。您正在编辑的脚本或类的名字出现在Java编辑器框架的选项卡上。选项卡左边的星号表示有未保存的变更。您可以打开 Java 编辑器中的若干文件并通过单击适当的选项卡来在它们之间移动。如果在此窗口中进行处理时出现代码问题，就会在受影响行附近显示出一个问题标记。另外，在Java编辑器中单击右键会显示出关于脚本的各种菜单选项。除此，它就与每个其他的脚本编辑器相像了。

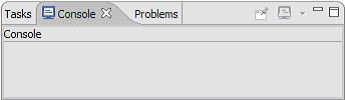
##### 2.1.5 Script Explorer



Test Perspective窗口右边窗格中的Script Explorer列出了脚本助手、助手超类或助手基类、测试数据库、验证点和当前脚本的测试对象。注意这些关于 Script Explorer的项：

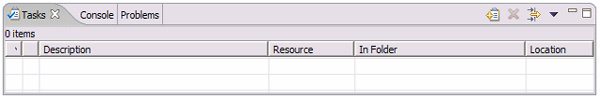
* Verification Points文件夹中包含所有为脚本记录的验证点。双击验证点会显示出 Verification Point Editor。
* Test Objects文件夹中包含了一列脚本可用的所有测试对象。该列中每个测试对象前面都有代表其作用的图标。双击 Test Object Map图标会显示测试对象地图。
* 右键单击验证点、测试对象地图或 Script Explorer中的测试资产显示出各种菜单选项。

##### 2.1.6 Console视图



Console视图显示来自脚本或应用程序的输出 -- 例如，System.out.print语句和未处理的Java异常。

##### 2.1.7 Tasks视图



Tasks视图显示错误、警告或其他由编译器自动生成的信息。该视图默认地列出工程中所有文件的所有任务，但您也许想要限定显示与当前脚本相关的任务。您可以通过点击Tasks视图标题中的 过滤器按钮来应用一个过滤器。

##### 2.1.8 状态栏

http://www.ibm.com/developerworks/cn/rational/r-rft/images/image043.jpg

Rational Functional Tester利用 Test Perspective窗口底下的状态栏来显示消息。

可参考<http://www.ibm.com/developerworks/cn/rational/r-rft/index1.html>

# 第三节：Rational Functional Tester产品使用

### 3.1 入门

IBM Rational Functional Tester，以下称作Functional Tester*，*是为那些需要对 Java、Microsoft Visual Studio.NET和基于Web的应用程序的测试进行高级控制的测试人员和 GUI 开发人员提供的高级自动化功能及回归测试工具。

Functional Tester提供两种用户界面选择：Microsoft Visual Studio和Eclipse。如果您的开发团队在 Visual Studio中构建应用程序，并且您的测试团队更习惯将 Visual Basic.NET 作为脚本语言，那么您可以在 VisualStudio IDE外壳中直接安装 Functional Tester，并在 Visual Basic.NET中获取测试脚本。

另一个选择是将Functional Tester安装到Eclipse平台上 —— 要么是您自己的Eclipse安装版，要么是 Rational提供的 Functional Tester安装包的一部分。有了 Functional Tester 的 Eclipse 版，测试脚本以Java代码形式获取。不管怎样，您都拥有完全环境的功能，以及商业标准脚本语言的通用性。

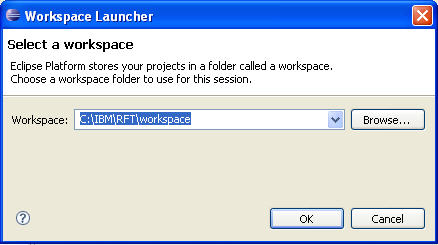
Functional Tester支持许多类型的应用程序的功能回归测试，包括那些用以下技术构建的应用程序：

* Java 平台
* Web（HTML）
* Microsoft® .NET
* Siebel
* SAPgui
* 基于终端的应用程序（3270、5250，及 VT100）

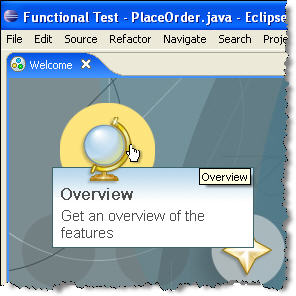
所有这些应用程序都可以用基于 Eclipse 的或基于 Visual Studio的 Functional Tester进行测试。虽然本教程的余下部分将使用基于Eclipse的版本，但是您在此处做的操作也能在基于 Visual Studio的版本中实现。

##### 3.1.1 **启动RFT**

通过选择Windows开始菜单的Programs>IBM Rational Functional Tester>IBM Rational Functional Tester>Java Scripting启动Functional Tester。当您第一次启动Functional Tester时，可能会提示您选择工作区位置，如图1所示。工作区可以位于您要存储工作产品的任何目录位置。如果在启动Functional Tester时让您选择工作区，那么您可以选择默认的，或新建一个自己的。

**图 1. Workspace launcher**  


在您选择了工作区之后，就显示出Welcome页面，如图2所示。它提供到教程和示例的快速链接。单击Overview标签，开始Functional Tester的漫游。

**图 2. Welcome 页面的 overview**   


关闭Welcome页面。Functional Test透视图出现。Eclipse中的透视*图*是某一特殊任务的工具和视图的集成。Functional Test透视图，如名字所示，提供着重于验证软件应用程序功能操作的开发人员或质量保证专家所需要的视图。

### 3.2 **准备测试环境**

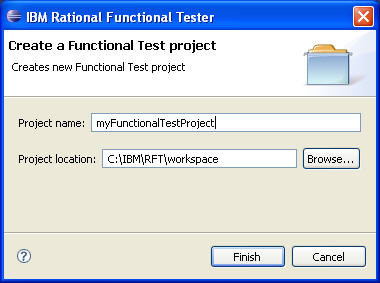
本部分将带您完成准备应用程序环境的过程，以便能使用 Functional Tester进行测试。您将：

* 创建存储库，或项目*，*其中存储您的Functional Tester工件
* 启用进行测试的应用程序环境
* 配置被测应用程序

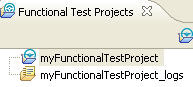
##### 3.2.1 **新建功能测试项目**

创建名为MyFunctionalTestProject的Functional Test 项目，用来存储您的测试脚本、数据池，和项目图。如果您还没有启动 Functional Tester 的话，启动它并选择前面部分中显示的工作区。

1. 如果您还没有启动 Functional Tester 的话，启动它并选择前面部分中显示的工作区。
2. 在工作台中选择File>New>Functional Test Project**。**您应该见到图 3 中的内容。
3. 输入MyFunctionalTestProject 作为项目名，并单击Finish**。**

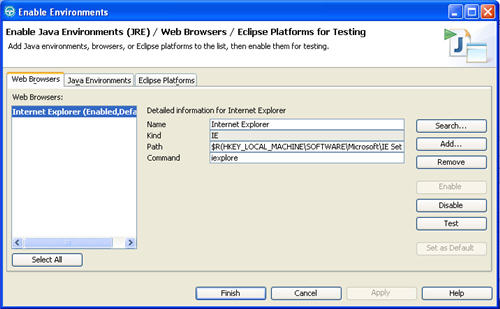
**图 3. Create Functional Test Project 向导**  


如您在图4中所见到的，Functional Test Projects视图中有两个新条目：一个用于显示测试工件，一个用于显示结果日志。

**图 4. Functional Test Projects 视图**  


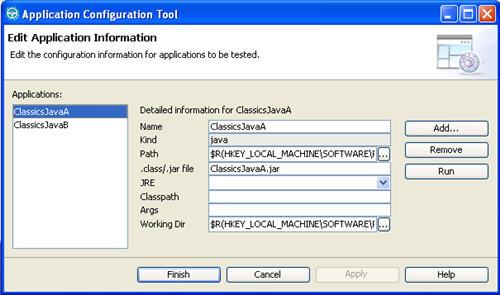
##### 3.2.2 **启用应用程序环境**

接下来，您需要启用应用程序运行时环境。这样做可以使 Functional Tester 能深入运行时环境，识别被测应用程序中的对象。

1. 选择Configure>Enable Environments for Testing**...**。您将看到Enable Environments窗口中的三个选项卡，如图5所示。这些是需要用到的三个类别或领域，它们是根据您所测试的应用程序的类型而定。一般情况下，Internet Explorer是测试回放的默认浏览器，并默认被启用。  
     
   **图 5. Enable Environments 窗口**  
   
2. Functional Tester 能够测试基于 Eclipse的插件及RCP应用程序。如果您已经安装了另外的Eclipse平台，那么您可以在Eclipse Platforms 选项卡中为测试启用它们。
3. 选择Java Environments选项卡。IBM SDP Java Runtime Environment（JRE）是用于测试Java应用程序的默认 JRE，您可以通过 Functional Tester启用它。如果您的应用程序使用不同的JRE，通过此界面来启用您的 JRE。对于本教程，使用该默认的 JRE 进行回放，所以在您在此处不用做任何改动。
4. 单击Finish关闭窗口。

##### 3.2.3 配置被测试应用程序

配置被测试应用程序实际上不影响或变更您的应用程序，只是在 Functional Tester 环境中创建一个快捷方式，或指针*，*使得启动应用程序变得更容易，并且使得测试对其他测试机来说更轻便。

1. 在 Application Configuration Tool窗口中选择Configure>Configure Applications for Testing...，如图 6 所示。为Functional Tester 测试而配置的所有应用程序的列表显示了出来。Functional Tester中出现称为 *ClassicsJava.*的示例Java Swing应用程序。提供 ClassicsJava的两个构建，两个都是在安装 Functional Tester 时自动定义的。您可以看到，在此处，如何指定运行参数，例如JAR文件，classpath、JRE，和工作目录。  
     
   **图 6. Application Configuration Tool 窗口**  
   
2. 因为您将使用本教程中的ClassicsJava示例应用程序，所以此时在该屏幕上您没有什么需要改动的。单击Finish关闭窗口。

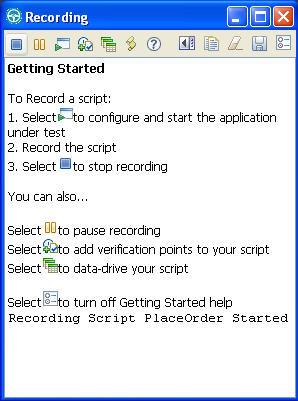
### 3.3 录制测试

通过手动编码、录制，或二者的组合可以创建测试。即使您打算做一些定制的编码，首先录制测试也比从头开始编写测试要更简单。

在本教程中，您将录制在示例ClassicsJava应用程序中购买CD的操作。

##### 3.3.1 启动录制器

1. 单击工具栏上的红色录制按钮，启动录制器。
2. 在 Record a Functional Test脚本向导中，输入脚本名称PlaceOrder，并单击Finish。测试录制器启动了，Functional Tester和录制工具栏最小化，如图7中所示。录制工具栏上有许多在录制过程中有用的特性。在本教程中只用到一些。

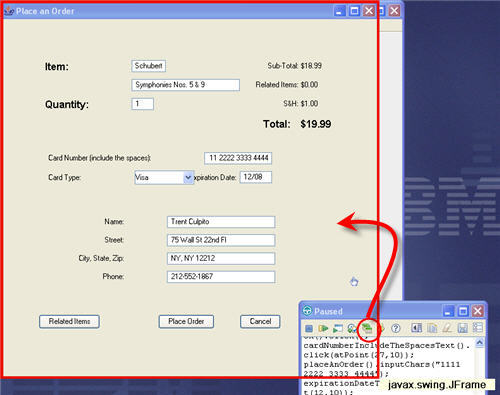
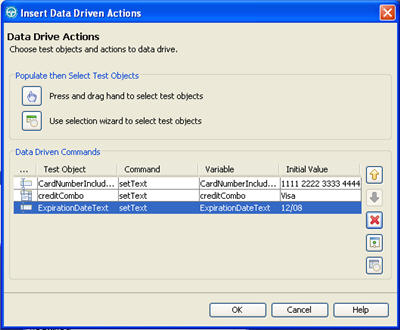
**图 7. 录制工具栏**  


##### 3.3.2 启动被测试的应用程序

1. 单击Start Application 按钮，以启动ClassicsJavaA - java应用程序快捷方式。ClassicsJava build A 启动了。
2. 牢记您此时在 ClassicsJava 应用程序中所做的一切都被录制下来。使用 Composers 树控件来选择Schubert > Symphonies Nos. 5 & 9。
3. 单击Place Order。
4. 此示例应用程序没有真正地检查登录或密码。为了简化，当向您请求这些内容时，接受默认值，但要选择Remember Password复选框。该复选框将在本教程的随后部分说明Functional Tester测试回放引擎的健壮性。
5. 单击OK，转到 Place an Order 窗口。

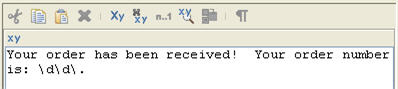
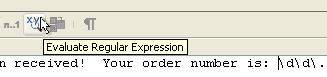
##### 3.3.3 创建数据驱动的订单

在测试脚本中默认获取所有的键盘输入。Functional Tester 拥有非常强大的功能：它可以将用户输入的数据与测试的过程和导航命令分开。这样做的优点是能够以各式各样数据的不同集合重复使用相同的测试过程，令您可以复用通用的测试，并且在极大减少您在创建反复的测试中涉及的时间和工作。

1. 在 Place an Order窗口中，输入Card Number 1111 2222 3333 4444，Card Type为 Visa，并输入Expiration Date 12/08。
2. 在录制工具栏上，将 Insert Data Driven Commands 图表拖到订单表单上，使得整个表单上包围红色方框，如图 8 所示。
3. **图 8. 选择用于数据驱动访问的字段**  
   
4. 如图9中的Insert Data Driven Actions 窗口出现了。Functional Tester默认您想要数据驱动面板上的所有测试对象或字段。然而，在这样的情况下，您只希望数据驱动信用卡信息。这使您使用同样的脚本来运行有多个信用卡的测试。要删除不必要的测试对象，选中一个，并单击红色 X 图表来删除。对所有的值都进行该操作，除了*CardNumberIncludeTheSpaces、creditCombo*，和*ExpirationDateText*。
5. **图 9. Insert Data Driven Actions 窗口**  
   
6. 单击OK确认选择。
7. 单击 Place an Order窗口上的 Place Order按钮。出现一个确认窗口。验证该窗口中的信息，如下部分中所示。

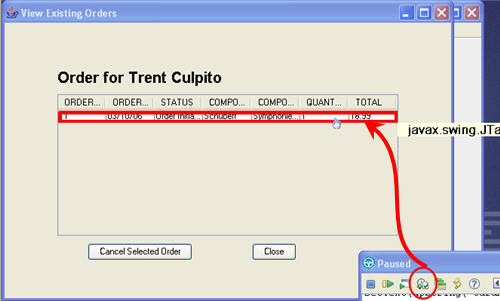
##### 3.3.4 验证动态数据

应用程序经常要响应您不能完全预料的数据和信息的输入。例如，在 ClassicsJava 中您将收到的确认信息包含二位的确认号码。您也许能够预料到这将是未来订单中的二位号码，但每次您运行该应用程序时该号码的值都是不同的。 您需要一种方法来验证消息的*模式*，而灵活处理该号码的实际值。Functional Tester 验证点具有这样的灵活性，如您将在本部分中所了解到的。

1. 从录制工具栏中将Insert Verification Point or Action 按钮拖到 Message 窗口中的文本上，如图 10 所示。当红框只圈住了以“Your order has been received...”开头的文本时，释放鼠标按钮。
2. **图 10. 选择要验证的标签**  
   
3. 存在许多种类型的验证点。在这里，您将添加一个简单的数据验证点。单击Next两次以接受默认选项。
4. 在向导的 Verification Point Data窗口上，单击 Convert Value to Regular Expression 按钮。正则表达式是一种形式的模式，匹配您可能已经熟悉的语言（参见[工具条](https://www6.software.ibm.com/developerworks/cn/education/rational/r-hellorft/section4.html#regex)，看看您在那里能找到更多信息）。
5. 在窗口中编辑数据，擦掉二位号码。
6. 在号码先前的位置上单击右键。上下文菜单提供一列最通用的正则表达式模式。使用上下文菜单在二位号码所在位置插入\d\d。该模式指出响应值应该是两位号码，但没有指定具体数字。如果号码是一个数字或三个数字，或者如果在每个数字位置上包含非 0-9 的内容，那么测试将失败。您的数据应该看起来类似图 11。  
     
   **图 11. 用正则表达式的验证点数据**  
   
7. Functional Tester 还为您提供了方法，验证您的正则表达式做没做您需要它做的事情。单击工具栏上的**Evaluate Regular Expression**，如图 12 中例举的。   
     
   **图 12. Evaluate Regular Expression 按钮**  
   
8. 在 Regular Expression Evaluator 中，您可以尝试 Match Against Value 框中的各种模式。变更取值，并按下Evaluate。您将看到您所输入的内容是否匹配正则表达式。
9. 当您的模式完成了您需要它做的事情，您对此感到满意时，单击Regular Expression Evaluator上的OK。
10. 在Verification Point 向导中单击Finish。这将您带回录制模式。
11. 在ClassicsJava 确认信息窗口中单击OK。

##### 3.3.5 验证静态数据

最后，您需要创建验证点来验证订单是否得到正确的处理。

1. 在 ClassicsJava菜单中，选择Order>View Existing Order Status**。**单击OK，再次登录。
2. 订单显示出来了，单击并将Verification Point and Action向导拖到订单信息上，以便订单外围圈上红框，如图 13 所示。  
     
   **图 13. 选择要验证的订单**  
   
3. 在向导的第一个窗口中单击Next。
4. 在第二个窗口中，选择数据值TableContents并单击Next。
5. 在最后的窗口中，单击Finish，将表格内容存储为基线数据，将来的测试运行将依据这些数据进行验证。这将您带回录制模式。

##### 3.3.6停止应用程序并停止录制

1. 在 View Existing Orders 窗口中单击 Close。
2. 单击ClassicsJava 窗口右上角的红X。
3. 单击Recording Toolbar上的Stop Recording。
4. Object Map and Help窗口出现了。此时，关闭这些窗口。

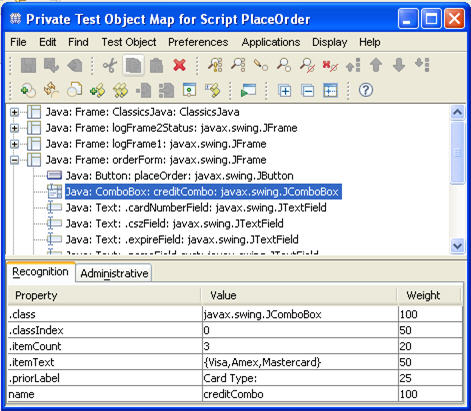
### 3.4 检验测试

##### 3.4.1 **检验测试对象图（Test Object Map）**

一旦录制完成，Functional Tester 便会自己复原。新录制的测试脚本位于中间的视图中。滚动浏览代码并观察所录制的命令。

右边是 Script Explorer 视图。该视图为您显示关于脚本的信息，例如类层次、数据池、验证点和与测试相关的测试对象。

双击 Test Objects 树上的creditCombo对象。Test Object Map 窗口打开了，如图14中所示。Test Object Map包含在录制过程中所收集的与所有遇到的对象相关的属性。该图中的属性将在回放的过程中用于确认应用程序中的对象。每一行右边的权重表示该属性对于识别有多重要。

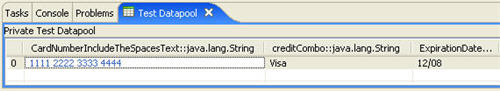
**图 14. 针对 creditCombo 的测试对象图**  


图中这个特殊的控件条目有两个属性。Functional Tester 专有的 ScriptAssure™模糊逻辑技术将使用所有属性来确定，在回放过程中每一项匹配得有多好。任何属性中的变更都不会导致回放失败。即使在新版的应用程序中有两个或三个属性发生变更，也不会导致回放失败。

关闭 Test Object Map。

##### 3.4.2 向数据池中添加数据变更

双击Script Explorer视图中的Private Test Datapool 对象。这样做打开了与位于窗口较下位置的脚本相关的数据池，如图15所示。

**图 15. Test Datapool 视图**  


1. 在 Test Datapool 中右键单击任意位置，并选择AddRecord**...**
2. 为您的测试显示出第二行数据。在信用卡号字段中双击，并输入新的卡号：1234 1234 1234 1234。
3. 将信用卡类型变更为Amex**。**
4. 在截至日期字段，输入截止日期06/08。

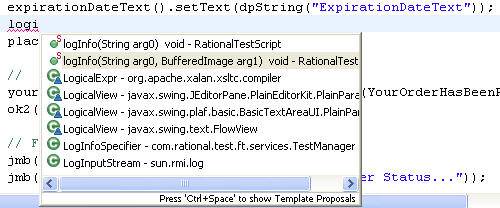
##### 3.4.3 向测试中添加定制操作

如早先所提到的，测试脚本完全是 Java 代码 ——不是 JavaScript，也不是专用语言。它给予您非常强大的功能来满足测试中的独特需求。而且，Functional Tester 提供丰富的 Application Programming Interface（API），通过这些 API，您可以访问测试对象，并控制测试的执行。

记住，在您录制测试的时候，首先您手动输入了信用卡号和截止日期，然后使用数据驱动命令向导来使 Functional Tester 在回放过程中用数据池中的内容填充这些字段。这样做在您的测试中遗留下一些不必要的命令。这实际上不会伤害到任何东西，但此时您可以删除它们，看看手动编辑脚本代码是如何进行的。

1. 在 PlaceOrder.java 视图中的测试脚本中，删除图 16 中显示的四行（注意到，如果您通过 Tab 键在字段间切换，而不用点击的方法，那么您的命令看起来会稍微有点不同）：

**图 16. 要删除的代码**  


1. 接下来，您需要添加一行代码，获取计算机屏幕的图像，放入测试日志中。在测试脚本中，找到用数据池中的值输入截止日期的命令。该行以 expirationDateText().setText开头。
2. 将您的光标放在该行的最后端，按下 Enter 键开始新的一行。
3. 开始输入logi，并按下Ctrl-Space。一个弹出窗口出现了，同时还出现了在该环境中有效的所有可能完成的代码，如图 17 所示。该特性通称为代码助手。
4. 按下键盘上的下箭头键，选中列表中的第二项。这是logInfo()，RationalTestScript 类中的静态方法，该方法在测试日志中创建了信息的输入，需要一个 String 类型的参数（标签）和一个BufferedImage（要插入的屏幕快照）。  
     
   **图 17. 代码助手入口**  
   
5. 按下 Enter 键，将该方法调用插入脚本中。
6. 选择String参数占位符。简单地输入"Screen Snapshot"（包括引号），并按下 Tab 键，前进到图像占位符。
7. 要获取屏幕图像，使用 root 测试对象的一个方法。输入getRootTestObject()。如果您需要的话，使用代码助手将输入工作减少到最少。
8. 在该行的末尾添加分号。保存您的变更。您的脚本应看起来像下面的图 18。

**图 18. 插入到脚本中的 logInfo() 调用**  
插入到脚本中的 logInfo 调用

在 Functional Tester 脚本中，您可以利用定制代码做很多很多事情。时间不允许我们进一步探究定制代码了，但类似扩展 API、强大的调试器，和代码助手的功能令定制脚本容易实现，甚至是对那些测试新手。

##### 3.4.4 **准备 ClassicsJava Build B**

Functional Tester 还对 ClassicsJava 示例应用程序进行第二次构建。在本教程的下一个部分中，您将依据此新的构建版本回放您的测试，以展示，随着应用程序的演进，Functional Tester 如何适应 UI 布局中的变更。但首先，您需要准备脚本来使用新的构建版本。

1. 定位到脚本中testMain()方法开头附近的startApp代码行。用ClassicsJavaB 替代 ClassicsJavaA。这样会导致 Functional Tester 使用不同的快捷方式来启动 ClassicsJava 的新版本。注意快捷方式名称区分大小写，因此务必使用大写*B。*
2. 保存变更。

### 3.5 运行自动化回归测试

在您开始执行回归测试之前，也许需要按顺序简要地回顾一下。这里有一些到目前为止您所做的事情：

* 通过创建一个测试存储库，设置测试环境，该存储库即项目*。*
* 您启用应用程序环境，让 Functional Tester 将 GUI 组件看成对象。
* 配置您的应用程序，以创建一个快捷方式，简化回放，并使您的测试更容易移植到其他系统。
* 然后，通过手动地使应用程序经过几个测试场景，对该测试进行录制。
* 在录制过程中，通过简单的向导，不用手动编码，将您所输入的数据与数据池联系起来。
* 您还插入了两个验证点。一个验证表格中的静态基线数据。另一个使用正则表达式利用模式匹配验证动态响应。
* 然后，将额外的数据集添加到数据池中，并添加定制代码使 API 调用在回放过程中记录屏幕快照，通过这种方式增强您的测试。

既然您已经配置好测试来启动被测应用程序的新构建版本了，那么您就准备执行功能回归测试并分析结果吧。

##### 3.5.1 执行测试

1. 在工具栏中，单击 **Run Functional Test Script**。回放向导为您提供在测试运行前指定日志名和选项的机会。接受默认的日志名，单击**Next**。
2. 在向导的 Playback Options 窗口中，您可以指定数据集的数字，测试可以通过该数字进行迭代。在 Datapool Iteration Count 框中，选择 **Iterate Until Done**。这表示您想让 Functional Tester 为数据池中的每一行执行一次。
3. 单击**Finish。** 静静地观察 Functional Tester 回放您测试的过程。注意，如果您之前运行过该测试，那么可能会出现提示让您确认是否想要覆盖日志。

在回放过程中，有几件要注意的事情：

* 注意，在 ClassicsJava 应用程序的这个版本中，UI 看起来有什么不同。一些对象被挪动或调整大小了，但 Functional Tester 能够将它们定位，并与它们恰当地交互。
* 在检查密码复选框之前，Functional Tester 在第一个登录框处暂停了几秒。当您分析日志文件中的测试结果时，您将进一步进行观察。
* 测试两次通过整个场景，对数据池中的每个数据集运行一次。
* 刚一完成，结果就显示在浏览器中的 HTML 日志中了。（当还原时，该浏览器窗口可能隐藏在 Functional Tester 窗口之后。）

在下一个部分中，您将分析日志文件中的结果。

### 3.6 分析测试日志中的结果

默认情况下，测试执行一完成，测试日志会立即默认打开。如果您已经关闭了浏览器，那么您可以在 Functional Test Projects 视图中的 myFunctionalTestProject\_logs 节点下找到测试日志。

日志的左侧框架提供了信息的快速导航。右侧边框提供所有细节。

##### 3.6.1 **ScriptAssure™**

滚动到第一个警告处，应如图 19 所示。消息指示“Object Recognition is weak (above the warning threshold)”。这是 UI 上的组件发生变更时的情况。事实上，在两个版本之间开发人员已经对该对象的多个属性进行了变更。在遇到这样的对象时，大多数测试自动化工具没有选择，只能报错，并放弃回归集的余下部分。这是一个实际的问题，如果您进行一个要运行若干小时的回归测试套件，并且您将其设置为整夜工作，然后，您第二天早上来到办公室，发现测试失败了，而您没得到显示失败的结果。

Functional Tester 的 ScriptAssure 能够识别出，这不是它在录制过程中看到的同一个组件。回放过程中的暂停是 Functional Tester 在等着看是否出现匹配的对象。在指定的超时时间过去之后，Functional Tester 确定屏幕上的对象与它所期望的足够接近，记录一个警告，并继续使用该对象。您可以配置许多属性权值、超时值，和阀值来定制 ScriptAssure 的行为，以满足您的需求。

**图 19. 测试日志中的 ScriptAssure 警告**  


##### 3.6.2 定制日志信息

在日志中滚动到下一个事件，名为“Screen Snapshot。”该事件是您所输入的定制代码的结果。单击 **Click to view full size** 的链接。该屏幕快照，如图 20 所示，是在回放过程中，作为您添加到测试中的logInfo() 命令的结果而被获取。注意所使用的信用卡号为“1111 2222 3333 4444”——来自于数据池第一行的值。

**图 20. 第一个屏幕快照**  

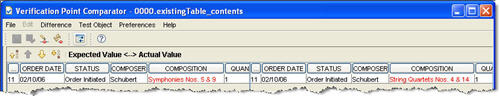

单击浏览器中的后退（Back）按钮，回到日志中。

##### 3.6.3 研究验证点状态

在日志中找到第一个验证点（VP）。测试日志中对于此验证点的条目将被称为“Verification Point [YourOrderHasBeenReceivedYou] passed，”并且应该类似图 21。您在此 VP 处使用了正则表达式来适应动态订单号。

**图 21. 通过 VP 日志事件**  


在日志中找到下一个验证点。该事件被称作“Verification Point [existingTable\_contents] failed”。单击**View Results** 链接，以更近一步调查此次失败。图 22 中所示的 Verification Point Comparator 以红色强调事实，VP 失败是因为，尽管订了 CD “Symphonies Nos. 5 & 9”，但是应用程序为“String Quartets Nos. 4 & 14”下了订单。您已经分析了 ClassicsJava 应用程序的版本 B 中的回归测试。一些对代码所做的看上去不相关的变更导致了在订购过程中使用了错误的内容。这是功能回归测试中的真正价值 —— 找到操作中无意识的变更。通过 Functional Tester 进行测试自动化，Functional Tester 实现了对每个版本都能进行回归测试的目标。

**图 22. Verification Point Comparator**  


关闭 Verification Point Comparator。

##### 3.6.4 验证数据池操作

继续在日志中滚动。您会注意到同组事件再次重复。当使用数据池中的第二行数据时，该迭代出现了。您可以通过打开第二个“Screen Snapshot”事件来确认数据池的操作。信用卡号是“1234 1234 1234 1234”，来自数据池第二行的值，如图 23 所示。

**图 23. 第二个屏幕快照**  


### 3.7 总结

本教程提供了对 Rational Functional Tester的入门介绍。您启用了应用程序环境，并录制了功能回归测试，还通过一些简单的定制编码对测试进行了增强。您还对应用程序新的构建版本执行了测试，并且分析测试结果，以揭示示例应用程序中回归测试缺陷。

可参考

<https://www6.software.ibm.com/developerworks/cn/education/rational/r-hellorft/index.html>

# 第四节：Rational Functional Tester产品好处

企业应用程序开发会涉及具有单独的小组，拥有清楚的职责的大型项目团队。团队常常被分成更小的分析人员团队、开发人员团队和测试人员团队。虽然这可以成为一个适当的管理结构，但它还创建了团队之间的屏障。我们的经验是当一个正式的测试团队成立时，开发人员可能更少进行自己代码的彻底测试了。这虽说不上是开发人员不关心开发出好的产品，但可以简单地说测试成为“其他人的工作”了。该问题常常混合了功能测试只是测试团队的工作，及单元测试只是开发团队的工作的假设。两级测试的界限是模糊的，并导致责任的间隙没有传达清楚。

重要的是您得承认开发人员（我们所知道的开发人员）喜欢解决复杂的设计和代码问题，而不喜欢开发测试。只有在以下的情况下，开发人员才认为测试是重要的：当测试与帮助解决问题相关；当测试能够确保问题能够被解决；当测试能够帮助解决工作中的其他问题。归根结底，本文将向您介绍可以使用 IBM® Rational® Functional Tester 来帮助解决这些问题的方法。它不是对更加传统的开发人员测试工具的替代，而是另一个能够使您更好地进行具体类型的测试的工具。

### 4.1关于开发者测试的公共问题

传统的开发人员测试在单元级别上发生。像 JUnit 和 NUnit 的工具会虑及开发人员功能的低级别操作，且它们提供用于报告通过或失败结果的标准报告结构。其他工具，如 JUnitBe，允许在一个网络容器中执行测试用例以测试容器资源。所有这些工具能够：

允许您用自己的母语和环境来书写测试用例

是可扩展的

包括多种用来实施您可能想要执行的大多数类型的测试扩展

然而，大多数这些工具不能从用户的观点来实施测试。那些从用户观点考虑测试的工具常常受限于一个具体的情况或环境（Swing、Internet Explorer，等等）。JFCUnit 是这种工具的一个杰出实例。另外，几乎所有这些工具都要求您手动书写测试用例代码。它们不提供获取测试、测试数据、或复杂情况的自动方法。由于它们依赖于要进行编码的测试用例、隐藏的接口和多种需要在不同环境中测试的工具，所以这些工具的使用对开发人员常常是有限制的。测试人员不会复用它们，除非那些测试人员本质上是技术性的。

开发人员常常处于紧张的项目进度中。这导致测试用例很难保持最新。当使用需要编码的解决方案时，如 NUnit，单元测试工件需要手动地与日期保持同步。如果您落后进度了，且测试用例打破了，那么您必须快速处理否则它将成为您开发工作史上仅仅的闪光点。常常只有最专注的开发人员会在面临紧张的期限时花时间来维护他们所有的单元测试。

用户界面表现出另外的难题。测试一个用户界面组件需要进行单元执行、数据输入和结果评估。复杂的界面（不论是基于 Web 的、Swing，或其他的）将问题扩大了。您可能需要测试组件对具体输入值的反应，但在到达该组件之前，需要对多个面板进行输入，才能到达此输入区域。在最终到达您所关心的要进行评估的值之前，这可能导致花费许多分钟来进行数据输入及面板显示。

### 4.2 Rational Functional Tester

Rational Functional Tester 是一个面向对象的自动测试工具，它可以使您测试多种应用程序。您可以通过记录对应用程序的测试来快速地生成脚本，并且您可以测试应用程序中的任意对象，包括对象的属性和数据。Rational Functional Tester 为您提供一个选择脚本语言和开发环境的机会 —— Eclipse 框架中的 Java™ 或者 Microsoft® Visual Studio ® .NET 开发环境中的 Microsoft ® Visual Basic® .NET。

将 Rational Functional Tester 集成到您的开发平台中可以把您带到一个新的效率级别上。这是一个巨大的优势，因为进行安装后，Rational Functional Tester 就成为您的本地开发环境中的一个无缝部分。Rational Functional Tester 提供记录和回放功能，并存储 Java 或 .NET 源代码的记录脚本。其允许您实际上不做任何工作就能创建可重复的测试脚本，也允许您使用您的程序设计技术用您本地的开发语言来增强脚本以满足具体的需求。

此功能及方便的使用为进行开发和测试团队之间的新级别的整合创造潜能。设想一个使用了包含典型的工作台应用程序工程的公共源存储库的工程，也包含测试工件工程。这里是几个此类型环境中的可能的情况：

情况 1：开发人员为系统增强录制新脚本并检入（check in）测试脚本。一个可以将脚本添加到一套适当的测试程序中并准备在开发人员结束编码之前就进行应用程序新版本的回归测试的测试人员立即得到了该脚本！

情况 2：开发人员为系统增强录制新脚本并检入（check in）测试脚本。一个可以使脚本成为数据驱动并开始向开发人员已经编码的简单用例中添加更多用例的测试人员立即得到了该脚本。

情况 3：开发人员收到了一张关于功能测试周期中遇到的问题的缺陷清单。缺陷清单标识遇到了问题的测试脚本。开发人员在本地调试模式下启动应用程序，在合适的位置设置断点，选择清单中标识的脚本并开始回放。

情况 4：测试人员正在开发针对产品某一特性的测试用例。测试人员对应用将要或应该表现的方式感到困惑。测试人员可以打开开发人员的测试脚本、检查代码和注释，并执行测试脚本以查看开发人员认为正在开发的内容是什么。如果测试人员看到问题，他可以以对于开发人员编码的新测试用例的形式来建议变更。

情况 5：开发人员坐下来维护不是他们开发的一小段代码。开发人员对代码表现的方式感到困惑。开发人员可以打开测试脚本，检查代码和注释，并运行测试脚本以查看原有开发人员所认为自己开发的内容是什么。

很难找到能够加强此级别上的开发人员与测试人员间的协作的工具。

不需要您的测试人员进行记录和重放

当您记录脚本的时候，Rational Functional Tester 会自动为被测的应用程序创建测试对象地图。对象地图包含每个对象的识别属性。当您更新对象地图中的识别信息时，任何使用该对象地图的脚本会共享更新的信息，这减少了维护的成本及整个脚本开发的复杂度。对象地图还提供向脚本添加对象的快速方法。它列出应用程序中用到的测试对象，不论它们是否为当前显示出的。您可以通过依据现有的地图或按照需要添加对象来创建新的测试对象地图。

Rational Functional Tester 还以 ScriptAssure 为特色：对象识别技术可以使您成功地回放脚本，甚至是当在测应用程序已经更新过的时候。您可以为测试对象必须通过的，能作为识别候选的识别分数设置门槛，并且如果 Rational Functional Tester 接受了分值高于指定门槛的候选者，您还可以向日志文件中书写警告。这减少了开发人员花在维护 GUI 级别的测试脚本（例如，不同于 JFCUnit）上的时间。

对象地图可以是共享的也可以是私有的。私有地图依附于脚本并且只由该具体的脚本进行访问，反之，共享地图由多个脚本共享。共享地图的优势是，当需要更新对象时，对地图中的一个更新就能修改多个脚本。在 Rational Functional Tester Projects 视图中或当您创建新的测试对象地图时，您可以将多个私有的或共享的测试对象地图合并成一个单个的共享测试对象地图。Rational Functional Tester 可以随意地更新您所选择的指向新合并的测试对象地图的脚本。

### 4.3超过简单的断言

在记录过程中，您可以向脚本中插入验证点以确定在被测应用程序的构造过程中对象的状态。验证点捕获对象信息（根据验证点的类型，可以是对象属性验证点或五种数据验证点类型中的一个 —— 菜单等级、 表格、文本、树形层次或 列表）并存储到基本数据文件中。此文件中的信息成为随后的构造过程中对象的期望状态。在执行测试之后，您可以利用 Verification Point Comparator 来分析区别，或者如果对象性能改变了，就更新基线（期望的对象状态）。其不仅能提供比传统的断言的更多种的功能，还将数据放在能复用的地方。

您可以用正则表达式替代识别属性 —— 或者考虑基于模式的识别，替代精确地匹配。这允许对象识别中的更多灵活性。您还可以将属性转变成正则表达式及 Verification Point Editor 中的或测试对象地图中的数值范围。另外 Regular Expression Evaluator 允许您在编辑时测试表达，这节省了您不得不运行脚本以查看模式是否运行的时间。

可参考<http://tech.ddvip.com/2008-12/122881524398669.html>

# 第五节：Rational Functional Tester产品资料

### 5.1学习

* 您可以参阅本文在 developerWorks 全球站点上的 [英文原文](http://www.ibm.com/developerworks/edu/i-dw-r-hellorft.html?S_TACT=105AGX52&S_CMP=cn-t-r) 。

地址：<http://www.ibm.com/developerworks/edu/i-dw-r-hellorft.html?S_TACT=105AGX52&S_CMP=cn-t->r

* 查看更多的“[Hello，World！](http://www.ibm.com/developerworks/cn/offers/lp/helloworld/)”系列教程。

地址：

<http://www.ibm.com/developerworks/cn/offers/lp/helloworld/>

* [IBM 面向服务的架构（Service Oriented Architecture，SOA）解决方案](http://www.ibm.com/cn/software/solution/soa/?S_TACT=105AGX52&S_CMP=dwcn-t-r) [（英文）](http://www.ibm.com/software/solutions/soa/?S_TACT=105AGX52&S_CMP=dwcn-t-r)：了解更多关于 IBM SOA 的信息。

地址：

<http://www-01.ibm.com/software/cn/solution/soa/?S_TACT=105AGX52&S_CMP=dwcn-t-r>

<http://www-01.ibm.com/software/solutions/soa/?S_TACT=105AGX52&S_CMP=dwcn-t-r>

* [Rational Functional Tester 产品专题](http://www.ibm.com/developerworks/cn/rational/products/functionaltester/)：找到关于 Rational Functional Tester 的技术文档、指导文章、教程、下载，和产品信息。

地址：

<http://www.ibm.com/developerworks/cn/rational/products/functionaltester/>

### 5.2获得产品和技术

* [Rational Functional Tester 试用版](http://www.ibm.com/developerworks/downloads/r/rft/?S_TACT=105AGX52&S_CMP=cn-t-r)：下载免费试用版

地址：

<http://www.ibm.com/developerworks/downloads/r/rft/?S_TACT=105AGX52&S_CMP=cn-t-r>

* [Extension for Terminal-Based Applications for Rational Functional Tester](https://www6.software.ibm.com/developerworks/cn/education/rational/r-hellorft/%20http:/www14.software.ibm.com/webapp/download/product.jsp?id=TMMS-62WPNR&s=z&cat=&S_TACT=105AGX19&S_CMP=tut)：下载免费试用版本。该扩展产品可以让您使用此处介绍的强大的 Functional Tester 功能来测试基于终端的应用程序。

地址：

<http://www14.software.ibm.com/webapp/download/product.jsp?id=TMMS-62WPNR&s=z&cat=&S_TACT=105AGX19&S_CMP=tut>

##### 5.3其他可参考学习网址

<http://cache.qihoo.com/wenda.php?do=snap&act=fetchHtmlsnap&m=0ee2e5440545a5f11c6051974820379f&surl=http%3A%2F%2Flive-free-or-die-hard.blogbus.com%2Flogs%2F25807614.html&kw=RFT%C8%EB%C3%C5>

<http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/com.rational.test.ft.help/com/rational/test/ft/ObjectNotFoundException.html>